ProtoDUNE analysis workshop
CERN, 27/01/2019

HighLAND analysis framework

Anselmo Cervera Villanueva
Alexander Izmaylov
Pablo Fernández

IFIC-Valencia

Requirements (my opinion)

- **Input files should be manageable:**
  - Small size
  - Fast to run over
  - Easy to understand for non-experts in simulation/reconstruction
- The analysis framework should be **decoupled** as much as possible from the much heavier **simulation/reconstruction** frameworks
  - Easy to install and compile
  - Easy to extend
  - Suitable for independent releases
- It should be possible to run the analysis and to do plots in a **laptop,** in Linux and MacOS
- It should be possible to run the analysis **without network connection**

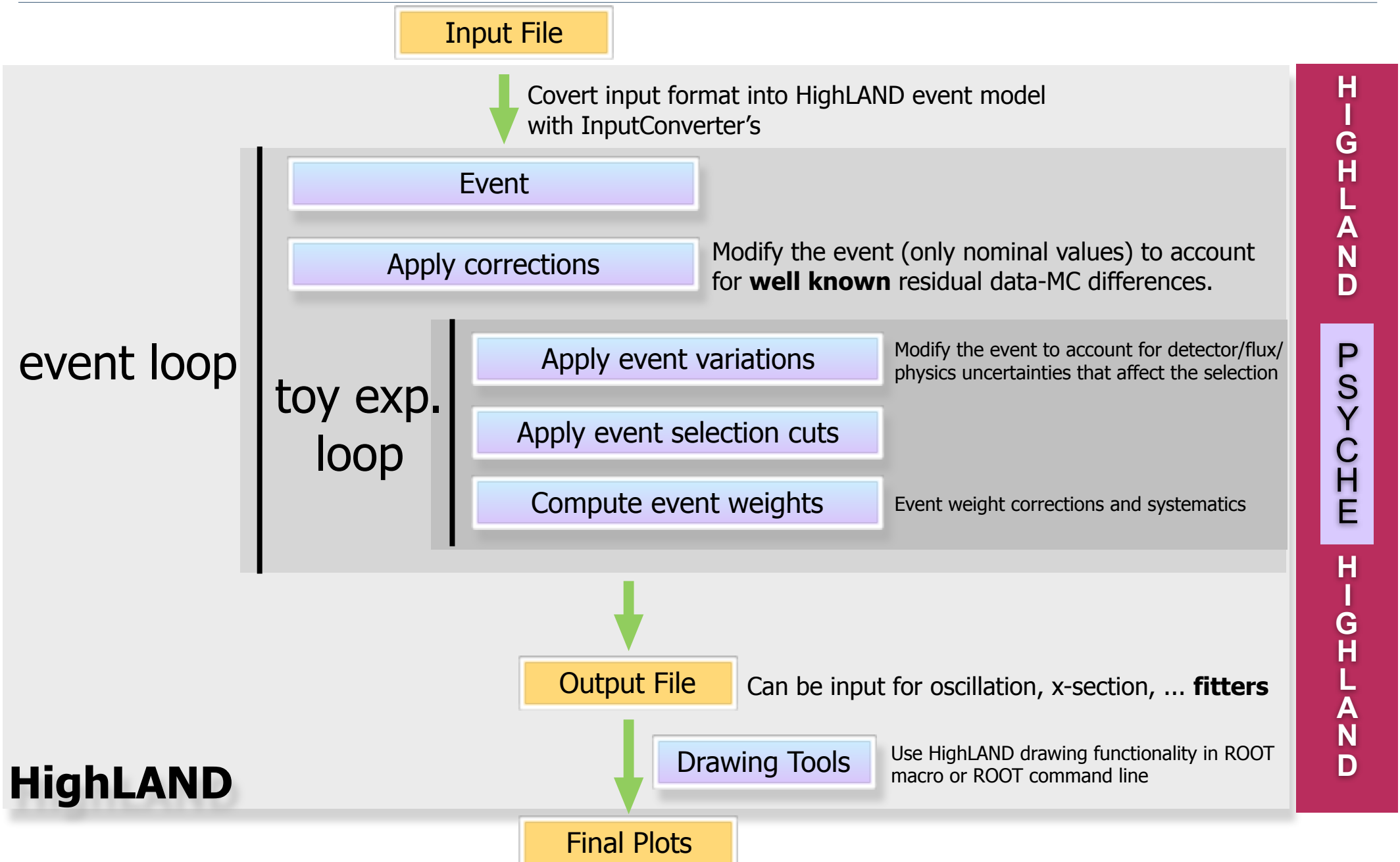
HighLAND analysis framework

- **HighLAND**
  - **High L**evel **A**nalysis at a **N**eutrino **D**etector
  - **High L**evel **An**alysis **D**evelopment
- HighLAND has been crucial for T2K near detector analyses
  - Has decreased considerably the learning curve and speed up analysis development
- **Highly optimized, thread safe, compiled c++ code** and run on the shell command line (not as root macro)
- **Very compact set of packages**: 1 minute to download and 5 minutes to compile
- Adapted to DUNE from the T2K near detector
  - A working prototype exists (see later)

What HighLAND provides

- **General analysis tools**
  - Event loop
  - Tools for multiple simultaneous event selections
  - Tools for numerical systematic error propagation
- Tools for **drawing** the analysis results
- **Event display**
- **Data Reduction** functionality. Example:
  - LArSoft --> MiniTree --> MicroTree --> NanoTree
- Tools for incorporating **specific analyses** into the framework
  - Extensible event data model
  - Hierarchy of analyses depending on each other

Analysis flow

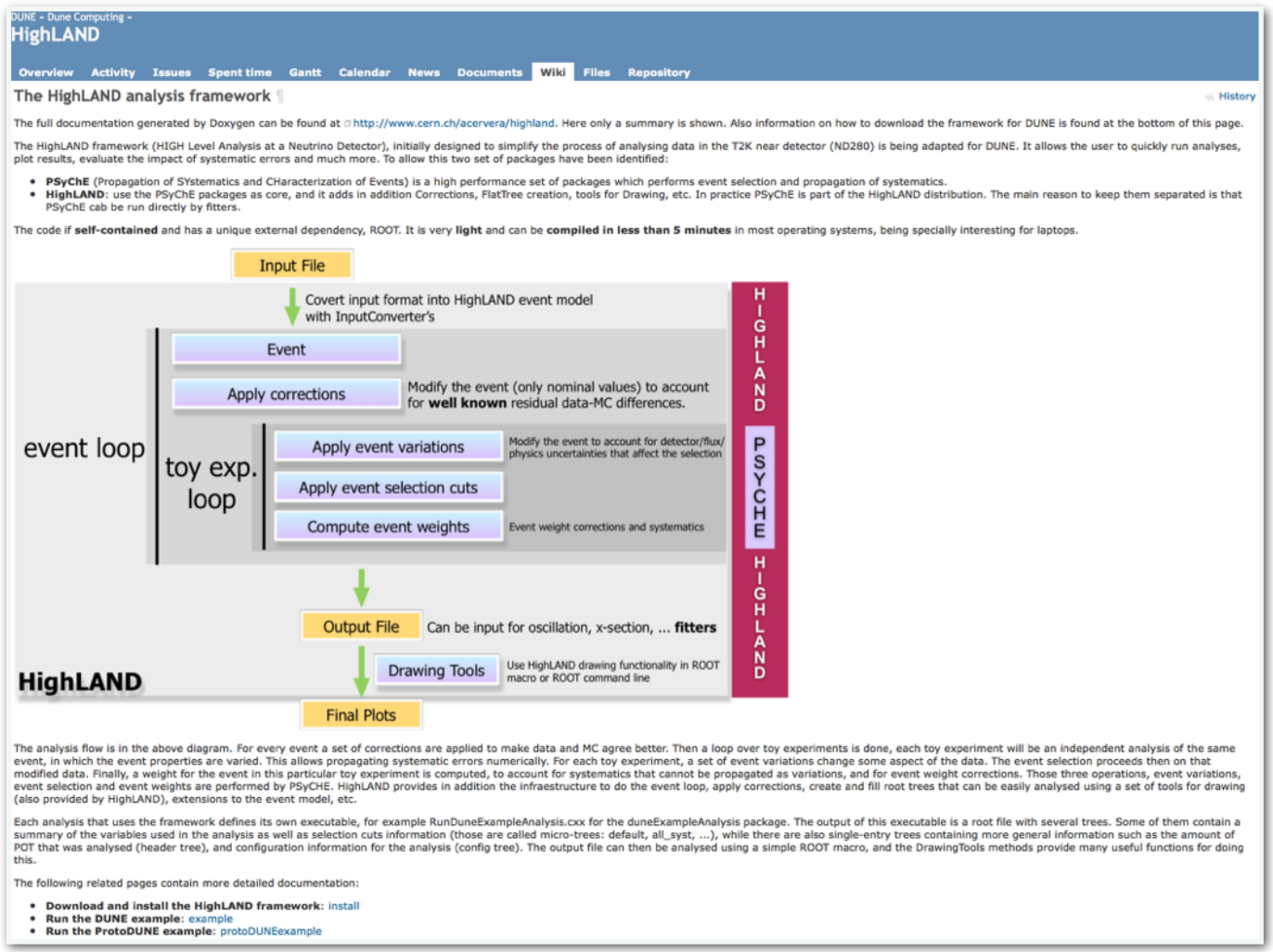


Elements and support

- **Repository**: A prototype version has been installed in a **gitLab** repository in Valencia
- **Building system:** To have a smooth transition from T2K we keep **CMT** for the moment. This is a very light package that can be obtained (automatically) from the same git repository
- **Bug tracking**: also redmine. Nothing there yet !!!
- **Validation**: **Jenkins** continues integration tests will be used at some point

Documentation

https://cdcvs.fnal.gov/redmine/projects/highland/wiki



DUNE » Dune Computing »

HighLAND

Overview | Activity | Issues | Spent time | Gantt | Calendar | News | Documents | Wiki | Files | Repository

The HighLAND analysis framework

History

The full documentation generated by Doxygen can be found at http://www.cern.ch/acervera/highland. Here only a summary is shown. Also information on how to download the framework for DUNE is found at the bottom of this page.

The HighLAND framework (HIGH Level Analysis at a Neutrino Detector), initially designed to simplify the process of analysing data in the T2K near detector (ND280) is being adapted for DUNE. It allows the user to quickly run analyses, plot results, evaluate the impact of systematic errors and much more. To allow this two set of packages have been identified:

- **PSyChE** (Propagation of SYstematics and CHaracterization of Events) is a high performance set of packages which performs event selection and propagation of systematics.
- **HighLAND**: use the PSyChE packages as core, and it adds in addition Corrections, FlatTree creation, tools for Drawing, etc. In practice PSyChE is part of the HighLAND distribution. The main reason to keep them separated is that PSyChE cab be run directly by fitters.

The code if **self-contained** and has a unique external dependency, ROOT. It is very **light** and can be **compiled in less than 5 minutes** in most operating systems, being specially interesting for laptops.

The analysis flow is in the above diagram. For every event a set of corrections are applied to make data and MC agree better. Then a loop over toy experiments is done, each toy experiment will be an independent analysis of the same event, in which the event properties are varied. This allows propagating systematic errors numerically. For each toy experiment, a set of event variations change some aspect of the data. The event selection proceeds then on that modified data. Finally, a weight for the event in this particular toy experiment is computed, to account for systematics that cannot be propagated as variations, and for event weight corrections. Those three operations, event variations, event selection and event weights are performed by PSyCHE. HighLAND provides in addition the infraestructure to do the event loop, apply corrections, create and fill root trees that can be easily analysed using a set of tools for drawing (also provided by HighLAND), extensions to the event model, etc.

Each analysis that uses the framework defines its own executable, for example RunDuneExampleAnalysis.cxx for the duneExampleAnalysis package. The output of this executable is a root file with several trees. Some of them contain a summary of the variables used in the analysis as well as selection cuts information (those are called micro-trees: default, all_syst, ...), while there are also single-entry trees containing more general information such as the amount of POT that was analysed (header tree), and configuration information for the analysis (config tree). The output file can then be analysed using a simple ROOT macro, and the DrawingTools methods provide many useful functions for doing this.

The following related pages contain more detailed documentation:

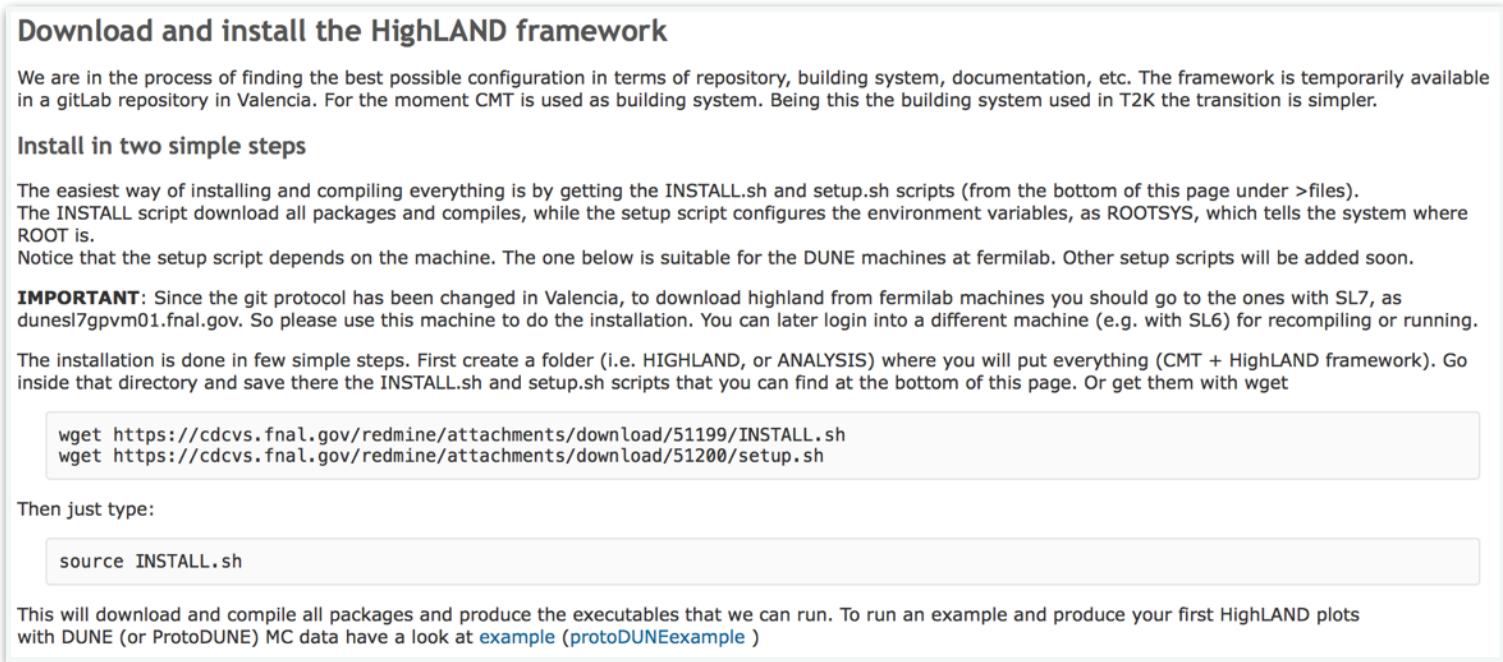
- **Download and install the HighLAND framework**: install
- **Run the DUNE example**: example
- **Run the ProtoDUNE example**: protoDUNEexample

- Includes a link to the Doxygen documentation

HighLAND installation

- This is an screen capture of

  https://cdcvs.fnal.gov/redmine/projects/highland/wiki/install



Download and install the HighLAND framework

We are in the process of finding the best possible configuration in terms of repository, building system, documentation, etc. The framework is temporarily available in a gitLab repository in Valencia. For the moment CMT is used as building system. Being this the building system used in T2K the transition is simpler.

Install in two simple steps

The easiest way of installing and compiling everything is by getting the INSTALL.sh and setup.sh scripts (from the bottom of this page under >files).
The INSTALL script download all packages and compiles, while the setup script configures the environment variables, as ROOTSYS, which tells the system where ROOT is.
Notice that the setup script depends on the machine. The one below is suitable for the DUNE machines at fermilab. Other setup scripts will be added soon.

IMPORTANT: Since the git protocol has been changed in Valencia, to download highland from fermilab machines you should go to the ones with SL7, as dunesl7gpvm01.fnal.gov. So please use this machine to do the installation. You can later login into a different machine (e.g. with SL6) for recompiling or running.

The installation is done in few simple steps. First create a folder (i.e. HIGHLAND, or ANALYSIS) where you will put everything (CMT + HighLAND framework). Go inside that directory and save there the INSTALL.sh and setup.sh scripts that you can find at the bottom of this page. Or get them with wget

```
wget https://cdcvs.fnal.gov/redmine/attachments/download/51199/INSTALL.sh
wget https://cdcvs.fnal.gov/redmine/attachments/download/51200/setup.sh
```

Then just type:

```
source INSTALL.sh
```

This will download and compile all packages and produce the executables that we can run. To run an example and produce your first HighLAND plots with DUNE (or ProtoDUNE) MC data have a look at example (protoDUNEexample)

- Please try, and if you have any problems email me (acervera@ific.uv.es) or submit an issue to **redmine**

  https://cdcvs.fnal.gov/redmine/projects/highland/issues

Status

- HighLAND was implemented for DUNE in 2016 and some analysis were done with MCC7. Talks at previous meetings:
  - **FD sim/reco 23/11/2015**: https://indico.fnal.gov/conferenceDisplay.py?confId=10882
  - **LBL 24/11/2015**: https://indico.fnal.gov/conferenceDisplay.py?confId=10861
  - **S&C 15/12/2015**: https://indico.fnal.gov/conferenceDisplay.py?confId=11030
  - **DUNE CM, 14/09/2016**, https://indico.fnal.gov/event/10613/session/18/contribution/52/material/slides/0.pdf
  - **PD meas/ana 13/10/2016**: https://indico.fnal.gov/event/13110/
  - **DUNE CM 24/01/2017**: https://indico.fnal.gov/event/10641/session/12/contribution/81/material/slides/0.pdf
- 2017-2018 busy years with ProtoDUNE hardware
- Since summer 2018 we are updating highland to work with root6
  - Minimal changes except in the reader of LArSoft files
  - The code compiles and runs
  - Main problem are the tools for drawing in an interactive root session:
    - because root cint have changed

HighLAND functionality

Reading the LArSoft file

- Using root TFile::MakeProject the **art** classes headers are recreated

```
recpack:cmt acervera$ ls ../../../LArSoftReader/v0r0/src/v07_02_00/
CRT__Hit.h
CRT__Trigger.h
LArSoftReaderLinkDef.h
LArSoftReaderProjectDict.cxx
LArSoftReaderProjectDict_rdict.pcm
LArSoftReaderProjectHeaders.h
LArSoftReaderProjectInstances.h
LArSoftReaderProjectSource.cxx
LArSoftTreeConverter.cxx
LArSoftTreeConverter.hxx
anab__Calorimetry.h
anab__CosmicTag.h
anab__FeatureVector_4_.h
anab__MVADescription_4_.h
anab__ParticleID.h
anab__T0.h
anab__cosmic_tag_id.h
art__Assns_raw__RawDigit_recob__Hit_void_.h
art__Assns_raw__RawDigit_recob__Wire_void_.h
art__Assns_recob__Cluster_recob__EndPoint2D_unsigned_short_.h
art__Assns_recob__Cluster_recob__EndPoint2D_void_.h
art__Assns_recob__Cluster_recob__Hit_void_.h
art__Assns_recob__Cluster_recob__Vertex_unsigned_short_.h
art__Assns_recob__Cluster_recob__Vertex_void_.h
art__Assns_recob__Hit_recob__SpacePoint_void_.h
art__Assns_recob__OpFlash_recob__OpHit_void_.h
art__Assns_recob__PFParticle_anab__T0_void_.h
art__Assns_recob__PFParticle_larpandoraobj__PFParticleMetadata_void_.h
art__Assns_recob__PFParticle_recob__Cluster_void_.h
art__Assns_recob__PFParticle_recob__PCAxis_void_.h
art__Assns_recob__PFParticle_recob__Shower_void_.h
art__Assns_recob__PFParticle_recob__SpacePoint_void_.h
art__Assns_recob__PFParticle_recob__Track_void_.h
art__Assns_recob__PFParticle_recob__Vertex_void_.h
art__Assns_recob__Shower_recob__Hit_void_.h
art__Assns_recob__Shower_recob__PCAxis_void_.h
art__Assns_recob__SpacePoint_recob__Hit_void_.h
art__Assns_recob__Track_anab__Calorimetry_void_.h
art__Assns_recob__Track_anab__CosmicTag_void_.h
art__Assns_recob__Track_anab__ParticleID_void_.h
art__Assns_recob__Track_anab__T0_void_.h
art__Assns_recob__Track_recob__Hit_recob__TrackHitMeta_.h
art__Assns_recob__Track_recob__Hit_void_.h
art__Assns_recob__Track_recob__SpacePoint_void_.h
art__Assns_recob__Track_recob__Vertex_void_.h
art__Assns_recob__Vertex_recob__Track_void_.h
art__Assns_recob__Wire_recob__Hit_void_.h
art__Assns_sim__AuxDetSimChannel_CRT__Trigger_void_.h
art__Assns_simb__MCTruth_simb__MCParticle_sim__GeneratedParticleInfo_.h
art__Assns_simb__MCTruth_simb__MCParticle_void_.h
art__BranchChildren.h
art__BranchDescription.h
art__BranchKey.h
art__BranchType.h
art__EDProduct.h
art__EventAuxiliary.h
art__EventID.h
art__FileFormatVersion.h
art__FileIndex__Element.h
art__HLTGlobalStatus.h
art__HLTPathStatus.h
art__Hash_2_.h
art__Hash_3_.h
art__Hash_5_.h
art__History.h
art__Parentage.h
art__ProcessConfiguration.h
art__ProcessHistory.h
art__ProductID.h
art__ProductProvenance.h
art__ProductRegistry.h
art__RNGsnapshot.h
art__RefCore.h
art__ResultsAuxiliary.h
art__RunAuxiliary.h
art__RunID.h
art__SubRunAuxiliary.h
art__SubRunID.h
art__Timestamp.h
art__Transient_art__BranchDescription__Transients_.h
art__Transient_art__ProcessHistory__Transients_.h
art__Transient_art__ProductProvenance__Transients_.h
art__TriggerResults.h
art__Wrapper_art__Assns_raw__RawDigit_recob__Hit_void___.h
art__Wrapper_art__Assns_raw__RawDigit_recob__Wire_void___.h
art__Wrapper_art__Assns_recob__Cluster_recob__EndPoint2D_unsigned_short___.h
art__Wrapper_art__Assns_recob__Cluster_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Cluster_recob__Vertex_unsigned_short___.h
art__Wrapper_art__Assns_recob__Hit_recob__SpacePoint_void___.h
art__Wrapper_art__Assns_recob__OpFlash_recob__OpHit_void___.h
art__Wrapper_art__Assns_recob__PFParticle_anab__T0_void___.h
art__Wrapper_art__Assns_recob__PFParticle_larpandoraobj__PFParticleMetadata_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Cluster_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__PCAxis_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Shower_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__SpacePoint_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Track_void___.h
art__Wrapper_art__Assns_recob__PFParticle_recob__Vertex_void___.h
art__Wrapper_art__Assns_recob__Shower_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Shower_recob__PCAxis_void___.h
art__Wrapper_art__Assns_recob__SpacePoint_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Track_anab__Calorimetry_void___.h
art__Wrapper_art__Assns_recob__Track_anab__CosmicTag_void___.h
art__Wrapper_art__Assns_recob__Track_anab__ParticleID_void___.h
art__Wrapper_art__Assns_recob__Track_anab__T0_void___.h
art__Wrapper_art__Assns_recob__Track_recob__Hit_recob__TrackHitMeta___.h
art__Wrapper_art__Assns_recob__Track_recob__Hit_void___.h
art__Wrapper_art__Assns_recob__Track_recob__SpacePoint_void___.h
art__Wrapper_art__Assns_recob__Track_recob__Vertex_void___.h
art__Wrapper_art__Assns_recob__Vertex_recob__Track_void___.h
art__Wrapper_art__Assns_recob__Wire_recob__Hit_void___.h
art__Wrapper_art__Assns_sim__AuxDetSimChannel_CRT__Trigger_void___.h
art__Wrapper_art__Assns_simb__MCTruth_simb__MCParticle_sim__GeneratedParticleInfo___.h
art__Wrapper_art__TriggerResults_.h
art__Wrapper_vector_CRT__Trigger___.h
art__Wrapper_vector_anab__Calorimetry___.h
art__Wrapper_vector_anab__CosmicTag___.h
art__Wrapper_vector_anab__FeatureVector_4_____.h
art__Wrapper_vector_anab__MVADescription_4_____.h
art__Wrapper_vector_anab__ParticleID___.h
art__Wrapper_vector_anab__T0___.h
art__Wrapper_vector_art__RNGsnapshot___.h
art__Wrapper_vector_larpandoraobj__PFParticleMetadata___.h
art__Wrapper_vector_raw__OpDetWaveform___.h
art__Wrapper_vector_raw__RawDigit___.h
art__Wrapper_vector_recob__Cluster___.h
art__Wrapper_vector_recob__EndPoint2D___.h
art__Wrapper_vector_recob__Hit___.h
art__Wrapper_vector_recob__OpFlash___.h
art__Wrapper_vector_recob__OpHit___.h
art__Wrapper_vector_recob__PCAxis___.h
art__Wrapper_vector_recob__PFParticle___.h
art__Wrapper_vector_recob__PointCharge___.h
art__Wrapper_vector_recob__Shower___.h
art__Wrapper_vector_recob__SpacePoint___.h
art__Wrapper_vector_recob__Track___.h
art__Wrapper_vector_recob__Vertex___.h
art__Wrapper_vector_recob__Wire___.h
art__Wrapper_vector_sim__AuxDetSimChannel___.h
art__Wrapper_vector_sim__OpDetBacktrackerRecord___.h
art__Wrapper_vector_sim__OpDetDivRec___.h
art__Wrapper_vector_sim__SimChannel___.h
art__Wrapper_vector_sim__SimPhotonsLite___.h
art__Wrapper_vector_simb__MCParticle___.h
art__Wrapper_vector_simb__MCTruth___.h
art__detail__AssnsBase.h
fhicl__ParameterSetID.h
geo__CryostatID.h
geo__PlaneID.h
geo__TPCID.h
geo__WireID.h
geo___plane_proj.h
geo___plane_sigtype.h
lar__range_t_unsigned_long_.h
lar__sparse_vector_float_.h
larpandoraobj__PFParticleMetadata.h
raw__OpDetWaveform.h
raw__RawDigit.h
raw___compress.h
recob__Cluster.h
recob__EndPoint2D.h
recob__Hit.h
recob__OpFlash.h
recob__OpHit.h
recob__PCAxis.h
recob__PFParticle.h
recob__PointCharge.h
recob__Shower.h
recob__SpacePoint.h
recob__Track.h
recob__TrackHitMeta.h
recob__TrackTrajectory.h
recob__Trajectory.h
recob__TrajectoryPointFlags.h
recob__Vertex.h
recob__Wire.h
sim__AuxDetIDE.h
sim__AuxDetSimChannel.h
sim__Chan_Phot.h
sim__GeneratedParticleInfo.h
sim__IDE.h
sim__OpDetBacktrackerRecord.h
sim__OpDetDivRec.h
sim__OpDet_Time_Chans.h
sim__SDP.h
sim__SimChannel.h
sim__SimPhotonsLite.h
simb__MCNeutrino.h
simb__MCParticle.h
simb__MCTrajectory.h
simb__MCTruth.h
simb___ev_origin.h
util__flags__BitMask_unsigned_int_.h
util__flags__Bits_t_unsigned_int_.h
util__flags__FlagSet_32_unsigned_int_.h
```

- So we don't need **art** nor **LArSoft** to read the LArSoft file

Reading the LArSoft file

- We read the “Events” tree in a LarSoft file

```
// General event info
eventsTree->SetBranchAddress("EventAuxiliary", &EventInfo);

// Reconstructed tracks
eventsTree->SetBranchAddress("recob::Tracks_pmtrackdc__Reco.", &Tracks);

// MC particles
eventsTree->SetBranchAddress("simb::MCParticles_largeant__G4.", &MCParticles);

// MC neutrinos
eventsTree->SetBranchAddress("simb::MCTruths_generator__GenieGen.", &MCNeutrinos);

// Reconstructed hits
eventsTree->SetBranchAddress("recob::Hits_lineclusterdc__Reco.", &Hits);

// Association between reconstructed hits and tracks
eventsTree->SetBranchAddress("recob::Hitrecob::Trackvoidart::Assns_pmtrack__Reco.", &Hits_Tracks);

// Channels
eventsTree->SetBranchAddress("sim::SimChannels_largeant__G4.", &SimChannels);
```

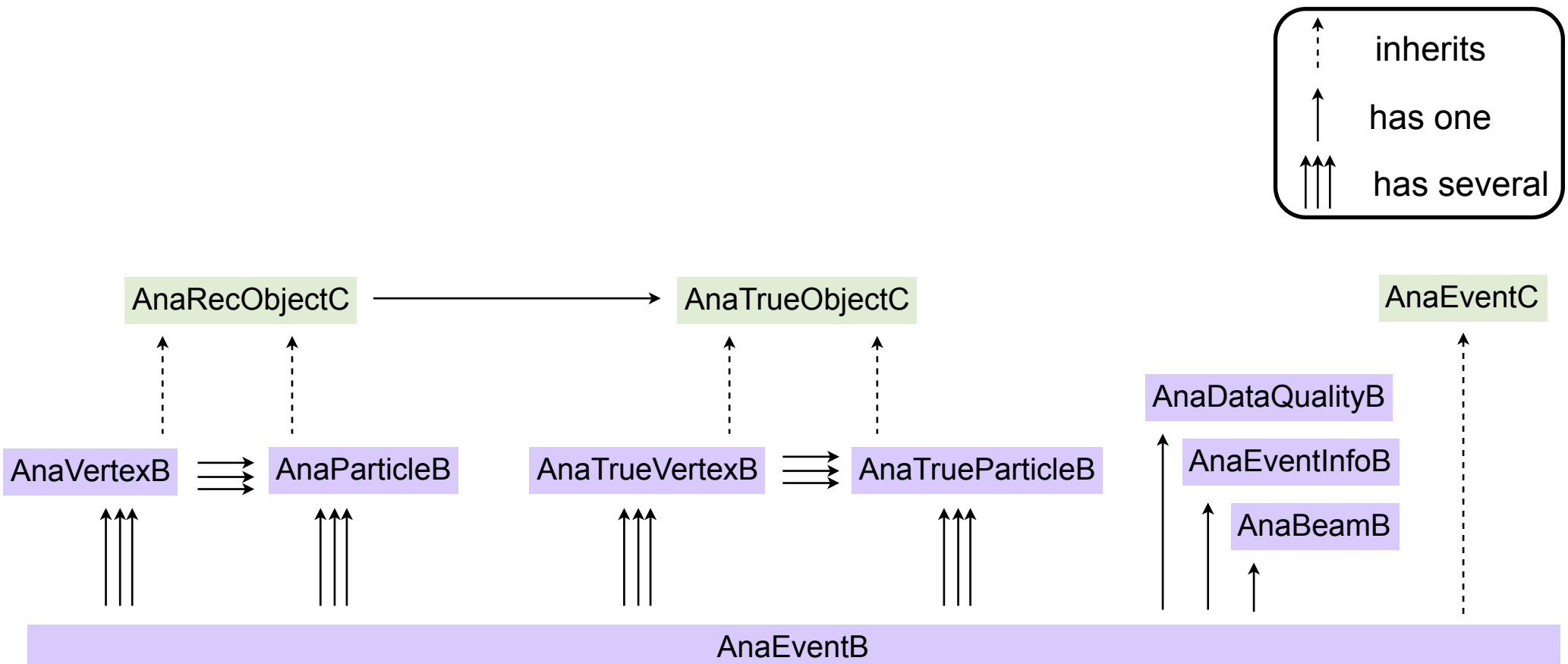
- Disable all branches we are not interested in to gain in speed

```
//-------------------- Disable the unnecessary branches ---
eventsTree->SetBranchStatus("art::*",0);
eventsTree->SetBranchStatus("sim::Beam*",0);
eventsTree->SetBranchStatus("sim::AuxDet*",0);
eventsTree->SetBranchStatus("sim::SimPhoton*",0);

eventsTree->SetBranchStatus("*shower*",0);

eventsTree->SetBranchStatus("raw*",0);
eventsTree->SetBranchStatus("anab*",0);
eventsTree->SetBranchStatus("recob::Wires*",0);
eventsTree->SetBranchStatus("recob::Vertexs*",0);
eventsTree->SetBranchStatus("recob::Space*",0);
eventsTree->SetBranchStatus("recob::Trackrecob*",0);
eventsTree->SetBranchStatus("recob::Tracks_emshower*",0);
```

```
eventsTree->SetBranchStatus("recob::Tracks_pmtrack_*",0);
eventsTree->SetBranchStatus("recob::Shower*",0);
eventsTree->SetBranchStatus("recob::Op*",0);
eventsTree->SetBranchStatus("recob::Hits_dcheat*",0);
eventsTree->SetBranchStatus("recob::Hits_gaushit*",0);
eventsTree->SetBranchStatus("recob::Hits_hitfd*",0);
eventsTree->SetBranchStatus("recob::Hitrecob::Space*",0);
eventsTree->SetBranchStatus("recob::Hitrecob::Wire*",0);
eventsTree->SetBranchStatus("recob::End*",0);
eventsTree->SetBranchStatus("recob::Cluster*",0);
//  eventsTree->SetBranchStatus("simb::MCTruths*",0);
eventsTree->SetBranchStatus("simb::MCFlux*",0);
eventsTree->SetBranchStatus("simb::GTruth*",0);
eventsTree->SetBranchStatus("simb::MCParticlesimb*",0);
```

HighLAND event model

- LArSoft info is extracted and saved into the HighLAND event model
- This is the current event model, which can be easily modified



Track and shower info

- This is the method to fill the AnaParticle's (track/shower)

```
//********************************************************************************
void LArSoftTreeConverter::FillParticleInfo(std::vector<AnaTrueParticleB*>& trueParticles, Int_t itrk, AnaParticle* part){
//********************************************************************************

  // The  ID of the particle (track or shower)
  part->UniqueID = Tracks->obj[itrk].fID;

  // General Particle ofinfo
  for (UInt_t j=0;j<3;j++){
    part->PositionStart[j]= Tracks->obj[itrk].fXYZ[0][j];
    part->DirectionStart[j]= Tracks->obj[itrk].fDir[0][j];
    part->PositionEnd[j]= Tracks->obj[itrk].fXYZ[Tracks->obj[itrk].fXYZ.size()-1][j];
    part->DirectionEnd[j]= Tracks->obj[itrk].fDir[Tracks->obj[itrk].fXYZ.size()-1][j];
  }

  // The number of hits
  part->NHits = Tracks->obj[itrk].fXYZ.size();
  SubDetId::SetDetectorUsed(part->Detector , SubDetId::kSubdet1_1);
  SubDetId::SetDetectorSystemFields(part->Detector);

  // Compute the track length
  part->Length = ComputeTrackLength(Tracks->obj[itrk]);

  // Compute the average dQdx of the reconstructed particle
  Int_t nsamples=0;
  for (UInt_t i=0;i<Tracks->obj[itrk].fdQdx.size();i++){
    for (UInt_t j=0;j<Tracks->obj[itrk].fdQdx[i].size();j++){
      part->AveragedEdx += Tracks->obj[itrk].fdQdx[i][j];
      nsamples++;
    }
  }
  part->AveragedEdx /= (Float_t)nsamples;

  // Associate a TrueObject to this Particle
  part->TrueObject = FindTrueParticle(itrk, trueParticles);
}
```

There is a method in LArSoft to do this. We had to reproduce that method inside HighLAND

Data Reduction functionality

- **LArSoft files are way too big**
  - I don't have much experience with them but getting them and running on them is not easy
  - Most information in those files is not needed for the analysis
- **The analysis should proceed on files that are manageable**
  - Small size
  - Fast to run over
  - Easy to understand for non-experts in simulation/reconstruction
- HighLAND can read LArSoft files and produce two other file types, much smaller and simpler
- Those files can be used as input for HighLAND analyses

Data Reduction functionality

- The event model can be dumped into a root file in two ways:
  - **MiniTree:** The class AnaEvent is saved
    - **Pros**: Does not need maintenance, changes en the event model are automatically propagated
    - **Cons:** Not very easy to navigate
  - **FlatTree:** Similar to the DUNE AnaTrees**.** The user decides the objects that are saved in the tree and the name of the variables. The output is a flat tree with basic type variables: double, float, int, char, and vectors of them.
    - **Pros**: Very easy to navigate
    - **Cons:** Difficult to maintain. Need to propagate changes in event model
- Both have similar size and running speed

Running directly on LArSoft files

- A list of LArSoft files in a text file: file.list

~50 GB for 230 events

```
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/93/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_2_20171229T053321_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/94/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_1_20171229T054519_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/95/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_8_20171229T054930_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/96/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_21_20171229T055045_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/97/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_66_20171229T054500_merged0.root
/pnfs/dune/tape_backed/dunepro/mcc10/mc/full-reconstructed/02/05/18/98/mcc10_protodune_beam_p3GeV_cosmics_3ms_sce_24_20171229T061658_merged0.root
```

- Run the example example over that list in dunegpvm03.fnal.gov

```
../Linux-x86_64/RunProtoDuneExampleAnalysis.exe -v -o test_230evt.root  file.list
```

- This is the final output on the screen

```
entry: 230 of 230 (100%) --> 230
time profile --------------
Ini Spill:         2321.84
Ini Bunch:         3.44837
Ini Conf:          0.577693
Ini Toy (v syst):  0.00119042
Ini Sel:           0.00724292
Selection:         0.0511615
End Sel (w syst):  8.36849e-05
End Toy:           0.0272403
End Conf:          4.85944
End Bunch:         0.183686
End Spill:         0.254492
Total:             2331.58
Total wo t:        2331.25
```

most time expended in reading the file

40 minutes for 230 events

MiniTree

- First create a MiniTree from the same list

```
../Linux-x86_64/RunCreateMiniTree.exe -v  -o mini_230evt.root ../../../protoDuneExampleAnalysis/v0r0/cmt/file.list

--------- time profile --------------
230 entries processed in 2291.63 seconds
```

takes 40 minutes

- The MiniTree is much smaller than the initial LArSoft files and can be easily transferred to a laptop

~20 MB (a factor 5000)

most space taken by truth info.
(More than 500 particles/event)

- Now running the analysis is much faster

```
../Linux-x86_64/RunProtoDuneExampleAnalysis.exe -v -o test_mini_230evt.root  ../../../highlandIO/v0r0/cmt/mini_230evt.root
```

```
entry: 230 of 230 (100%) --> 230
time profile --------------
Ini Spill:         2.01702
Ini Bunch:         0.0323527
Ini Conf:          0.00446773
Ini Toy (v syst): 0.000124216
Ini Sel:           7.7486e-05
Selection:         0.0019443
End Sel (w syst): 4.50611e-05
End Toy:           0.00111413
End Conf:          0.0445073
End Bunch:         0.0177789
End Spill:         0.0364242
Total:             2.1564
Total wo t:        2.15586
```

2 seconds for 230 events (a factor 1000)

Summary of data reduction

	LArSoft	MiniTree	Factor
File size	50 GB	20 MB	5000
processing time	2200''	2''	1000

- The MiniTree does not contain yet all info needed for the analysis but I don't think it will be much larger. Current info:
  - True Particles and True Vertices
  - All reconstructed tracks and showers, and the link to the corresponding True Particle.
  - No hits

Event Selection
Drawing tools

ProtoDUNE example

- An example exists in the git repository: **protoDuneAnalysisExample** package. Documentation exists in redmine. The package contains:
  - stoppingKaonSelection
  - dEdxCorrection, dEdxVariation (systematic)
- Selection of 1 GeV/c beam kaons stopping in the detector
- It was tested using MCC7

Event Selection

- A selection is a collection of steps (cuts or actions)
- Each selection inherits from **SelectionBase**, which has a main mandatory method **DefineSteps**

```
//*********************************************************************
void stoppingKaonSelection::DefineSteps(){
//*********************************************************************

  // Steps must be added in the right order
  // if "true" is added to the constructor of the step,
  // the step sequence is broken if cut is not passed (default is "false")
  AddStep(StepBase::kCut,    "> 0 tracks",         new AtLeastOneTrackCut());
  AddStep(StepBase::kAction, "find true vertex",   new FindTrueVertexAction_proto());
  AddStep(StepBase::kAction, "find main track",    new FindMainTrackAction());
  AddStep(StepBase::kAction, "find vertex",        new FindVertexAction());  // action from duneExampleAnalysis package
  AddStep(StepBase::kCut,    "kaon range",         new KaonRangeCut());
  AddStep(StepBase::kCut,    "> 1 track",          new MoreThanOneTrackCut());
  AddStep(StepBase::kCut,    "PIDA",               new PIDACut());
```

stoppingKaonSelection

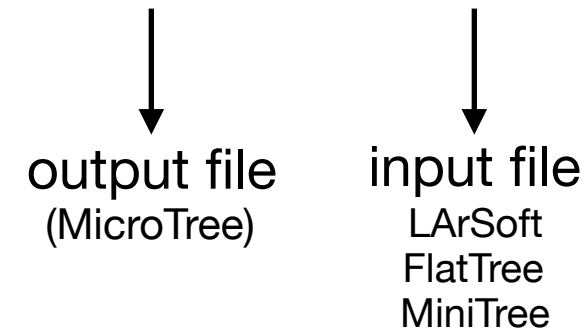
- Each step inherits from **StepBase** and implements the method **Apply**

```
//*************************************************
bool KaonRangeCut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//*************************************************

  // Cast the ToyBox to the appropriate type
  ToyBoxDUNE& box = *static_cast<ToyBoxDUNE*>(&boxB);
  if (!box.MainTrack) return false;

  Float_t length = static_cast<AnaParticle*>(box.MainTrack)->Length;
  if (fabs(length-200)<10) return true;
  else return false;
}
```

Running and plotting the example

- To run the example

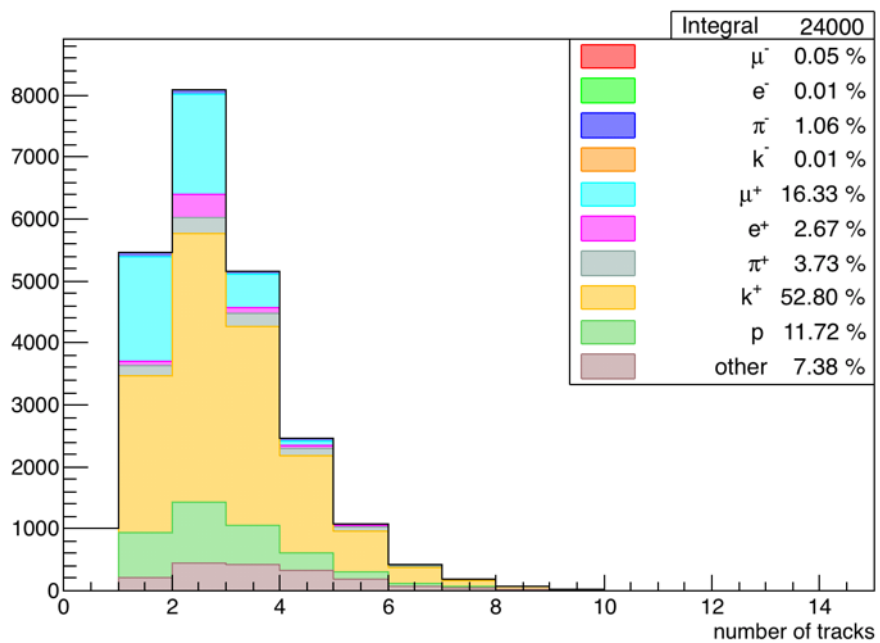
RunProtoDuneAnalysisExample.exe -o *kaon_1gev.root* *input.root*




```
Anselmo-Cerveras-MacBook-Pro-II:cmt acervera$ root -l kaon_1gev.root
root [0]
Attaching file kaon_1gev.root as _file0...
root [1] DrawingTools draw("kaon_1gev.root")
root [2] draw.DumpCuts()
------------------------------------------------------------------------
     Cuts for selection 'stoppingKaonSelection'  with no branches
------------------------------------------------------------------------
  #: index  type       title                    break  branches

  0: -1     cut        > 0 tracks               0      0
  4: -1     cut        kaon range               0      0
  5: -1     cut        > 1 track                0      0
  6: -1     cut        PIDA                     0      0
------------------------------------------------------------------------
```

Track multiplicity

- About 95% of events have at least one reconstructed track
- 1-4 reconstructed tracks is typical
- When >0 tracks, color indicates true particle associated with kaon candidate track in the event (see next)

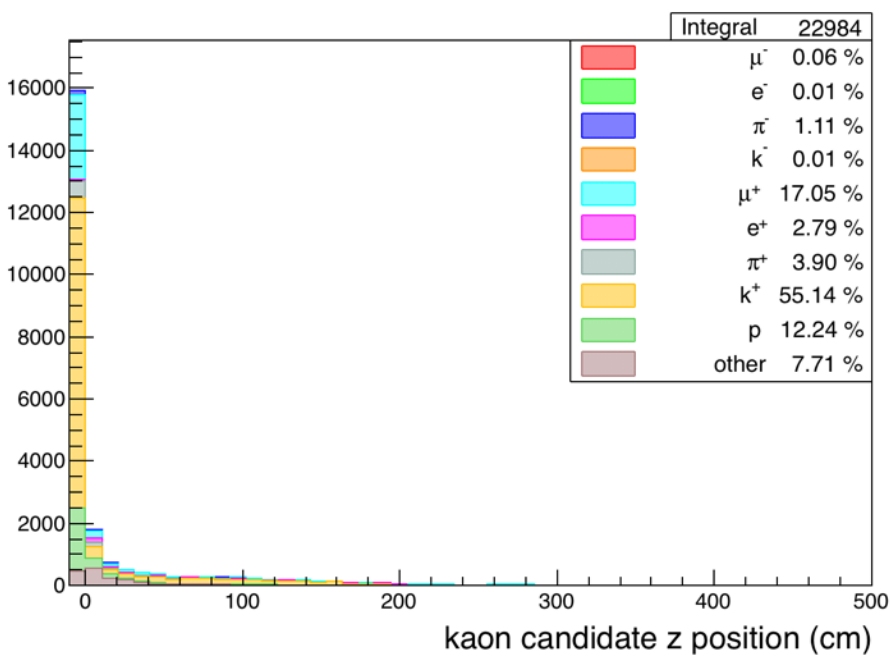


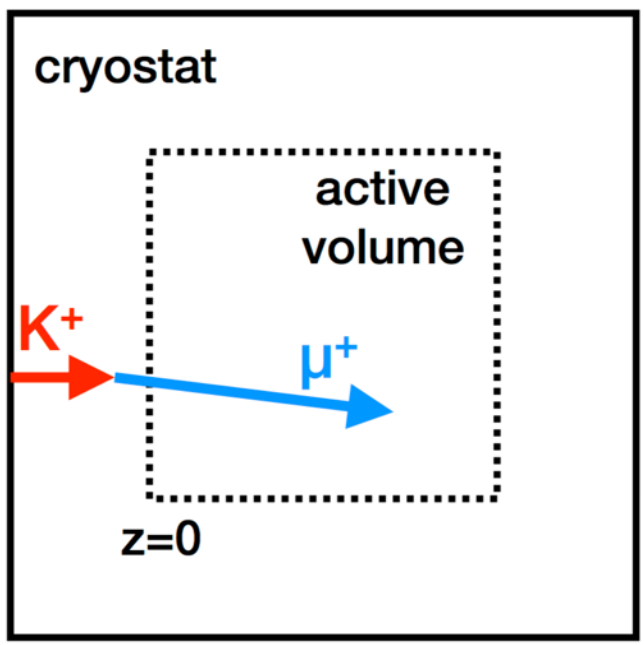
root commands to do the plot

```
draw.SetTitleX("number of tracks");
draw.Draw(default,"ntracks",15,0,15,"particle","accum_level>-1","HIST","DRAWALLMC PUR");
```

Kaon candidate track

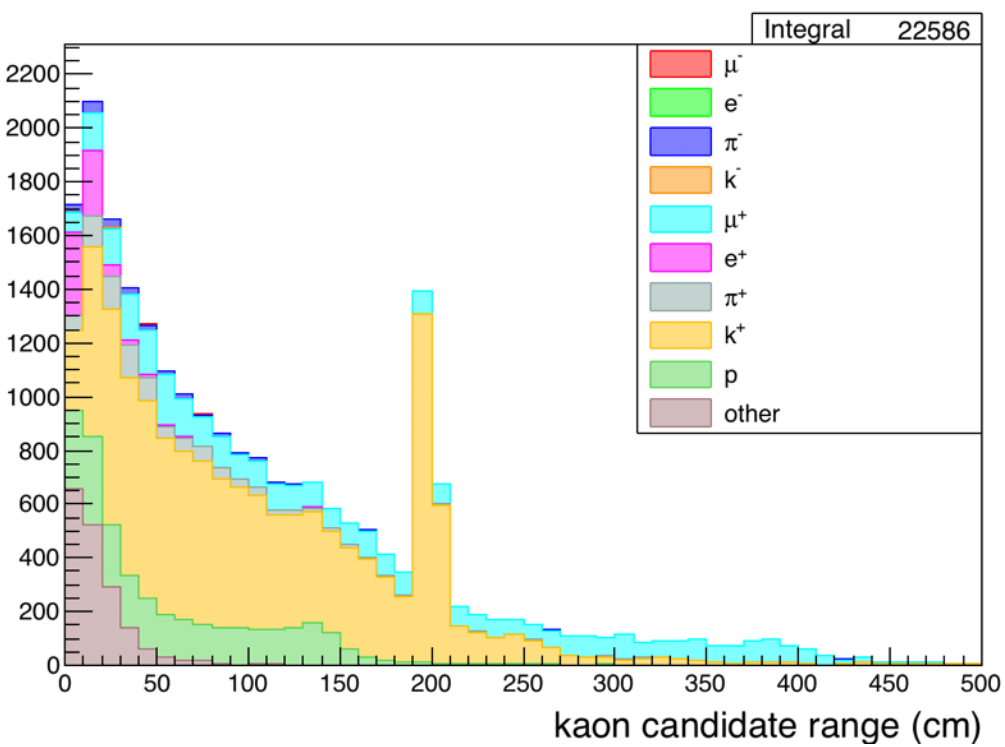
- Kaon candidate track defined as track starting most upstream (lowest z)
- With this definition, kaon candidate associated to true kaon only ~50% of the times
  - Partly because kaon decays or interacts before reaching the active volume (z>0) in ~30% of the simulated events

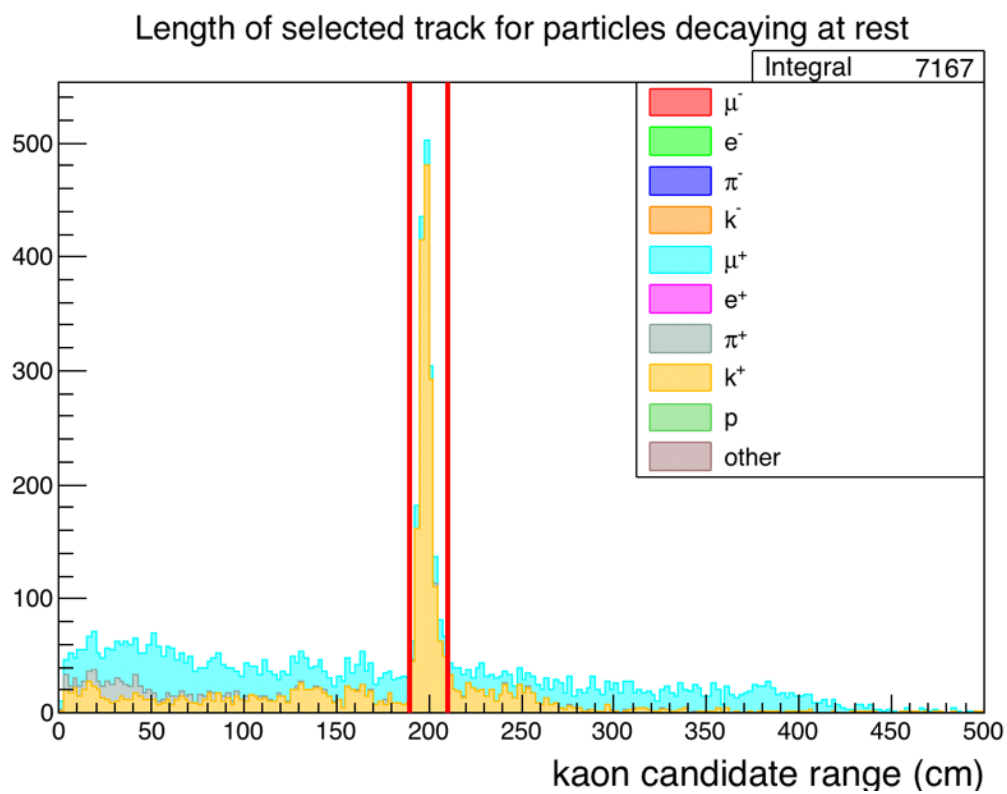




Fraction of stopping kaons and range cut

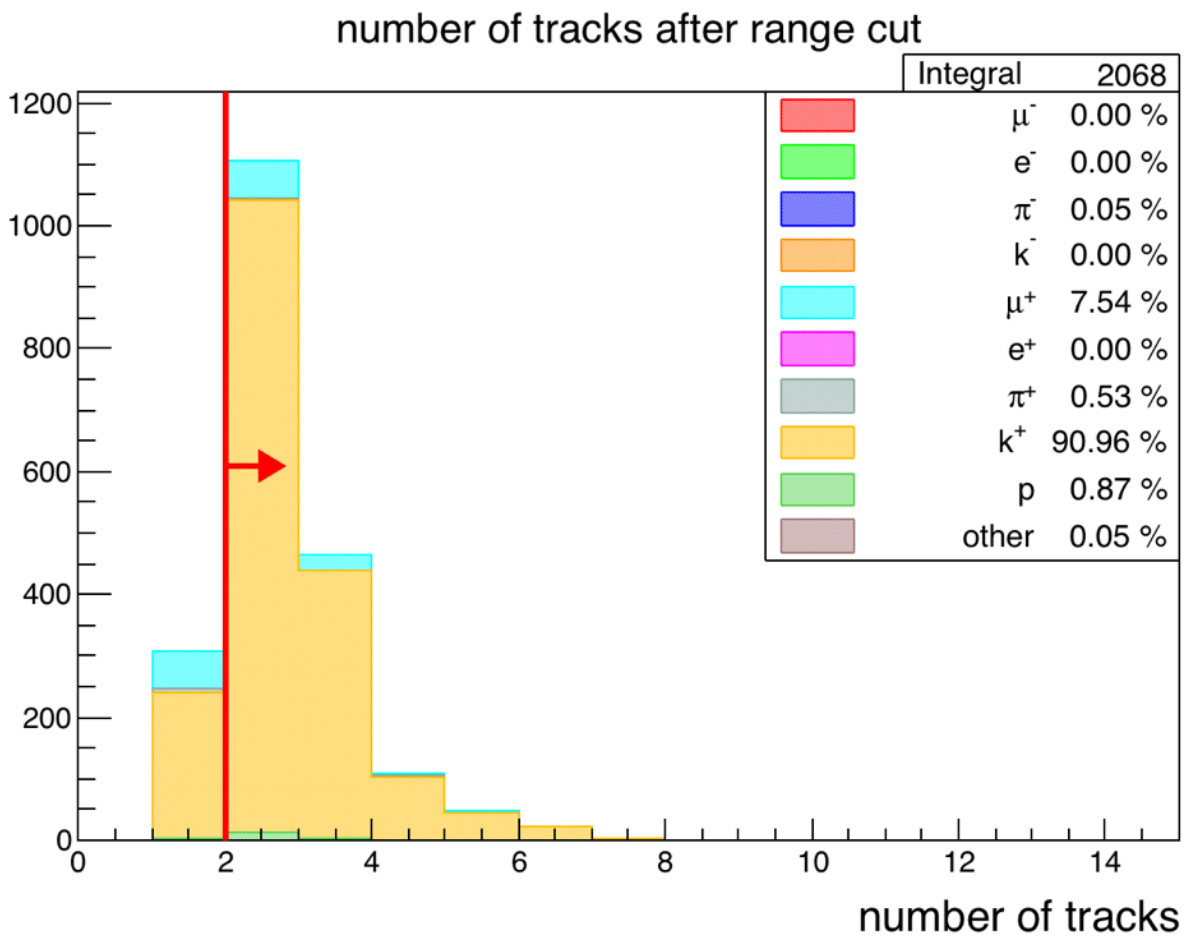
- Only ~10% of the events have a kaon stopping in the active volume
- Require ~200 cm range tracks to select all correctly reconstructed kaons decaying at rest





Multiplicity cut

- In addition to track range, >1 track requirement further improves kaon decay at rest selection
- Reason: tracks from kaon daughters are expected

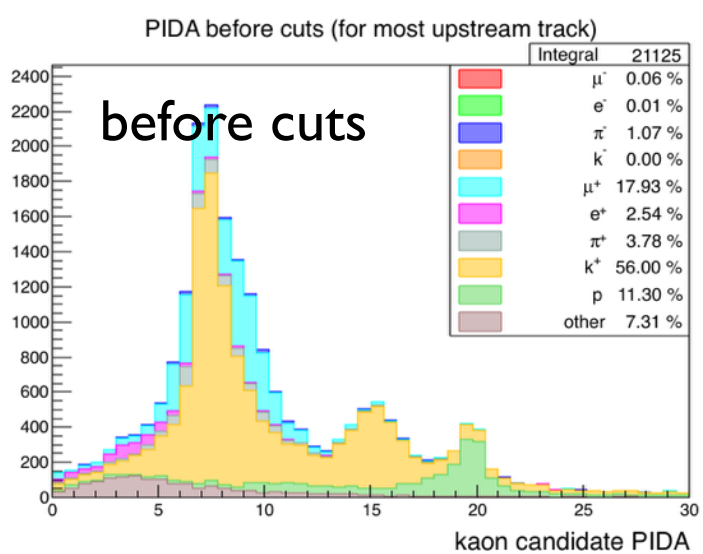


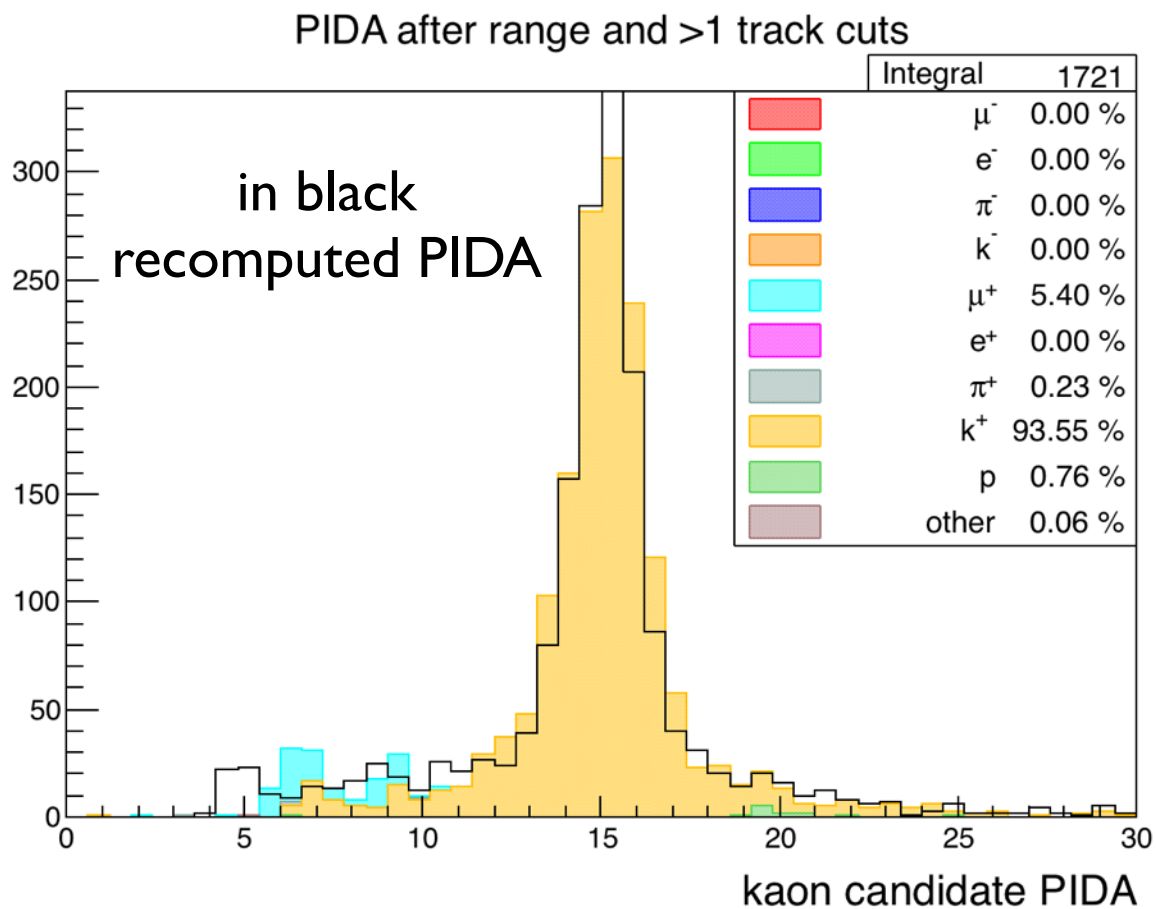
The PIDA variable

Averaged over all hits with residual range R<30

$$PIDA =< A_i >=< (dE/dx)_i R_i^{0.42} >$$

```
//********************************************************************
Float_t anaUtils::ComputePIDA(const AnaParticleB &track) {
//********************************************************************

  Float_t cut=30;

  Float_t PIDA=0;
  Int_t ncontrib=0;
  for (Int_t i=0;i<3;i++){
    for (Int_t j=0;j<track.NHitsPerPlane[i];j++){
      if (track.ResidualRange[i][j]<cut && track.ResidualRange[i][j]>0){
        ncontrib++;
        PIDA += track.dEdx[i][j]*pow(track.ResidualRange[i][j],0.42);
      }
    }
  }
  if (ncontrib>0) PIDA /= ncontrib*1.;

  return PIDA;
}
```



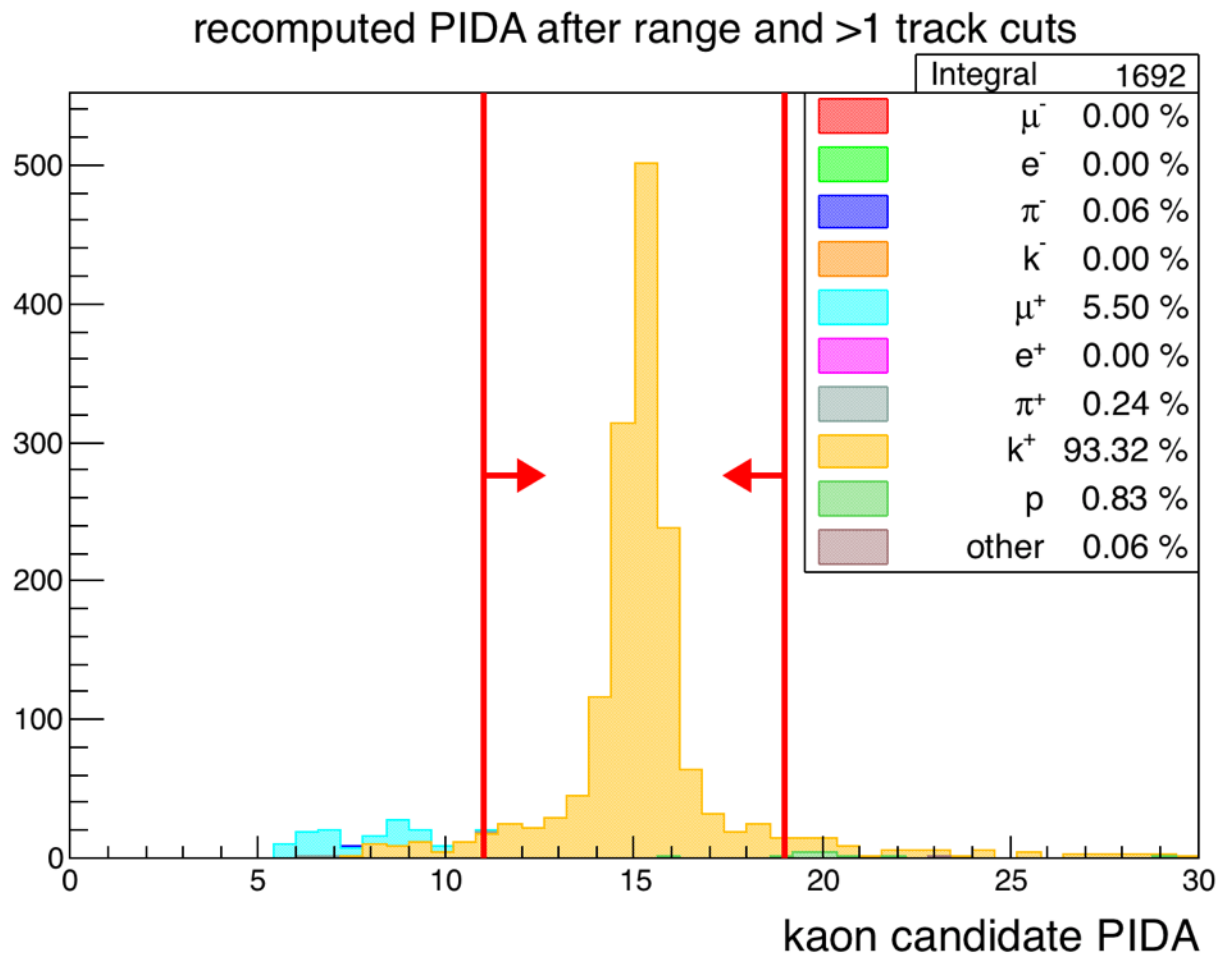


The recomputed PIDA is narrower because we have used all 3 wire plane while the one in the AnaTree only use one

The PIDA cut

the code for the cut

```
//**************************************************
bool PIDACut::Apply(AnaEventC& event, ToyBoxB& boxB) const{
//**************************************************

  (void)event;

  // Cast the ToyBox to the appropriate type
  ToyBoxDUNE& box = *static_cast<ToyBoxDUNE*>(&boxB);
  if (!box.MainTrack) return false;

  Float_t pida = anaUtils::ComputePIDA(*box.MainTrack);
  if (fabs(pida-15.)<4) return true;
  else return false;
}
```

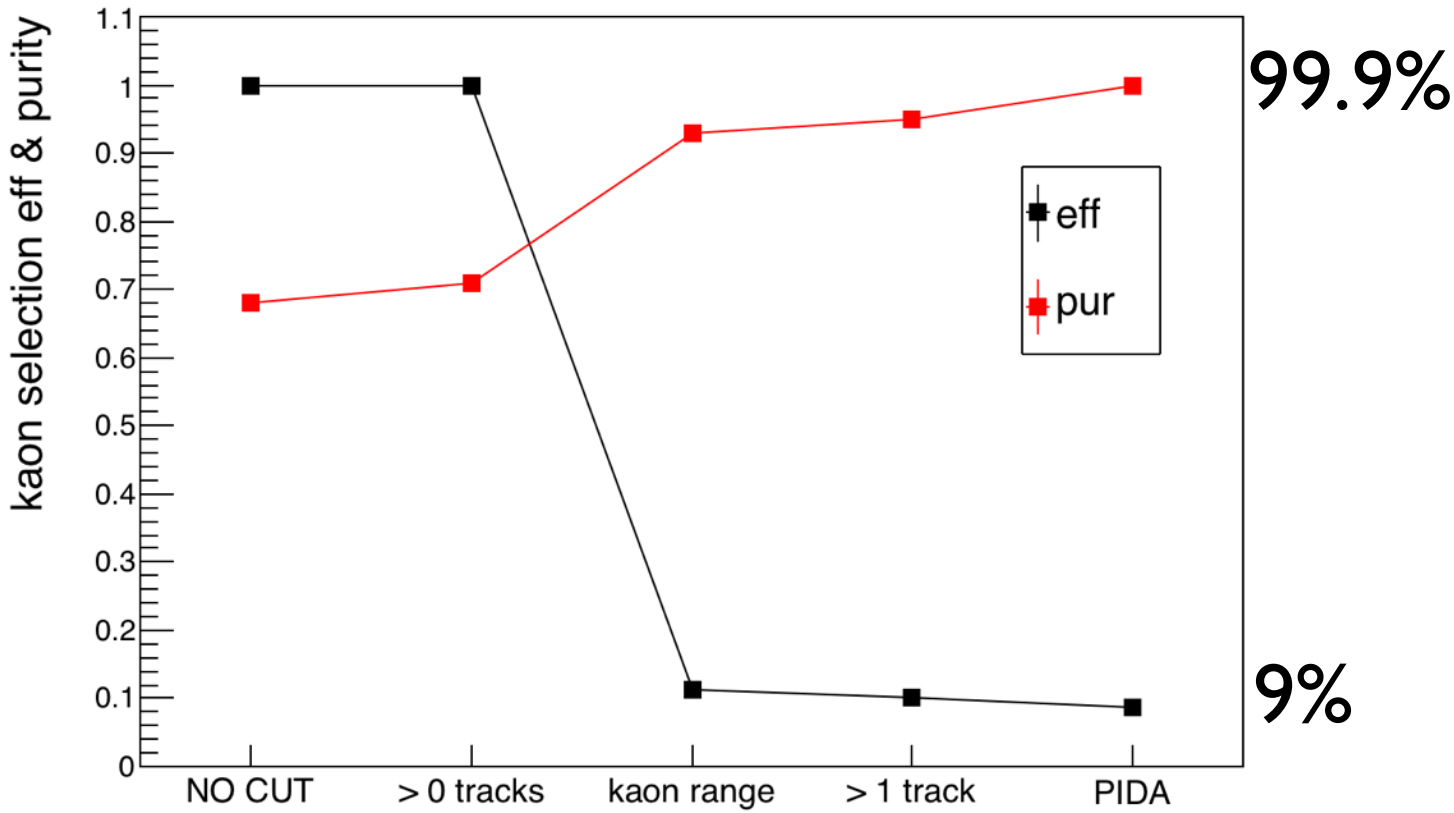


root commands to do the plot

```
draw.SetTitle("recomputed PIDA after range and >1 track cuts");
draw.SetTitleX("kaon candidate PIDA");
draw.Draw(default,"seltrk_pida_raw",50,0,30,"particle","accum_level>2","","PUR");
draw.DrawCutLineVertical(15.-4,true,"r");
draw.DrawCutLineVertical(15.+4,true,"l");
```

accum_level>2 means events passing cut 2
(>0 tracks, range cut, >1 track)

Efficiency and purity



root commands to do the plot

```
// Create a data sample instance with the micro-tree file.
// Needed to plot efficiency (from truth tree) and purity (from default tree) simultaneously
DataSample mc("test.root");

// kaon selection Efficiency and purity after each cut
draw.SetTitleY("kaon selection eff & purity");
draw.DrawEffPurVSCut(mc,"true_signal==1");
```

Additional functionality:

Corrections,

Systematics,

Event display,

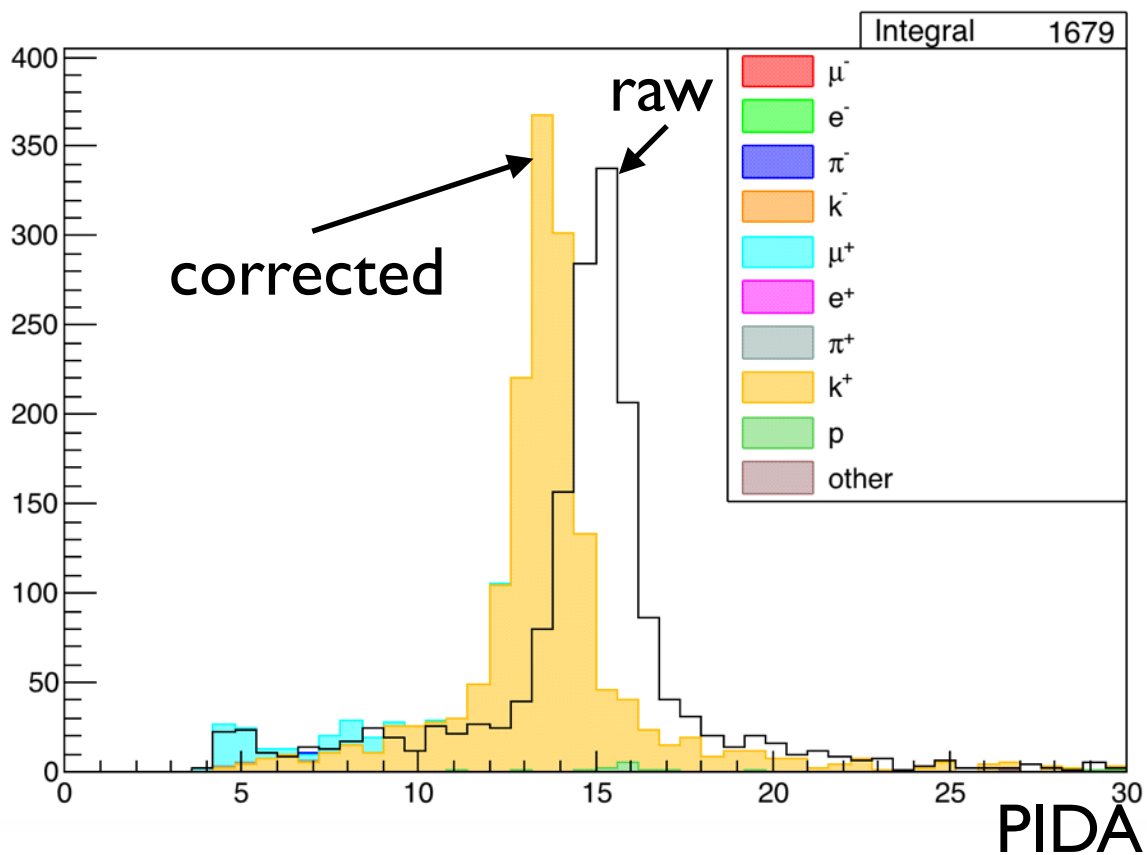
Analysis hierarchy

Corrections

- Correct a well known data MC difference to reduce the corresponding systematic
- Example: **dEdxCorrection**
  - Scales the dEdx of each hit by the correction factor and recomputes PIDA

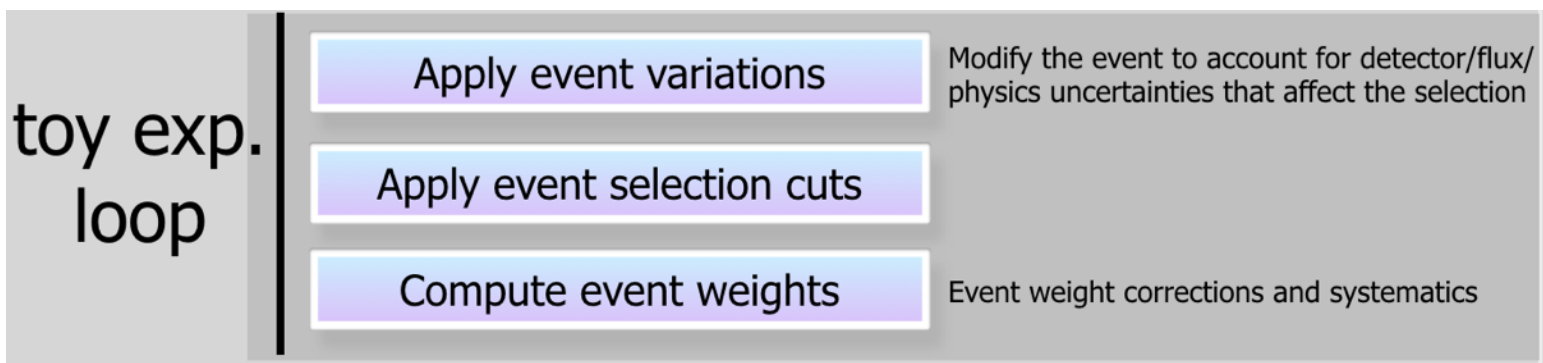
data/dEdx.dat

bins of PDG		correction	error on correction
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02



Systematics

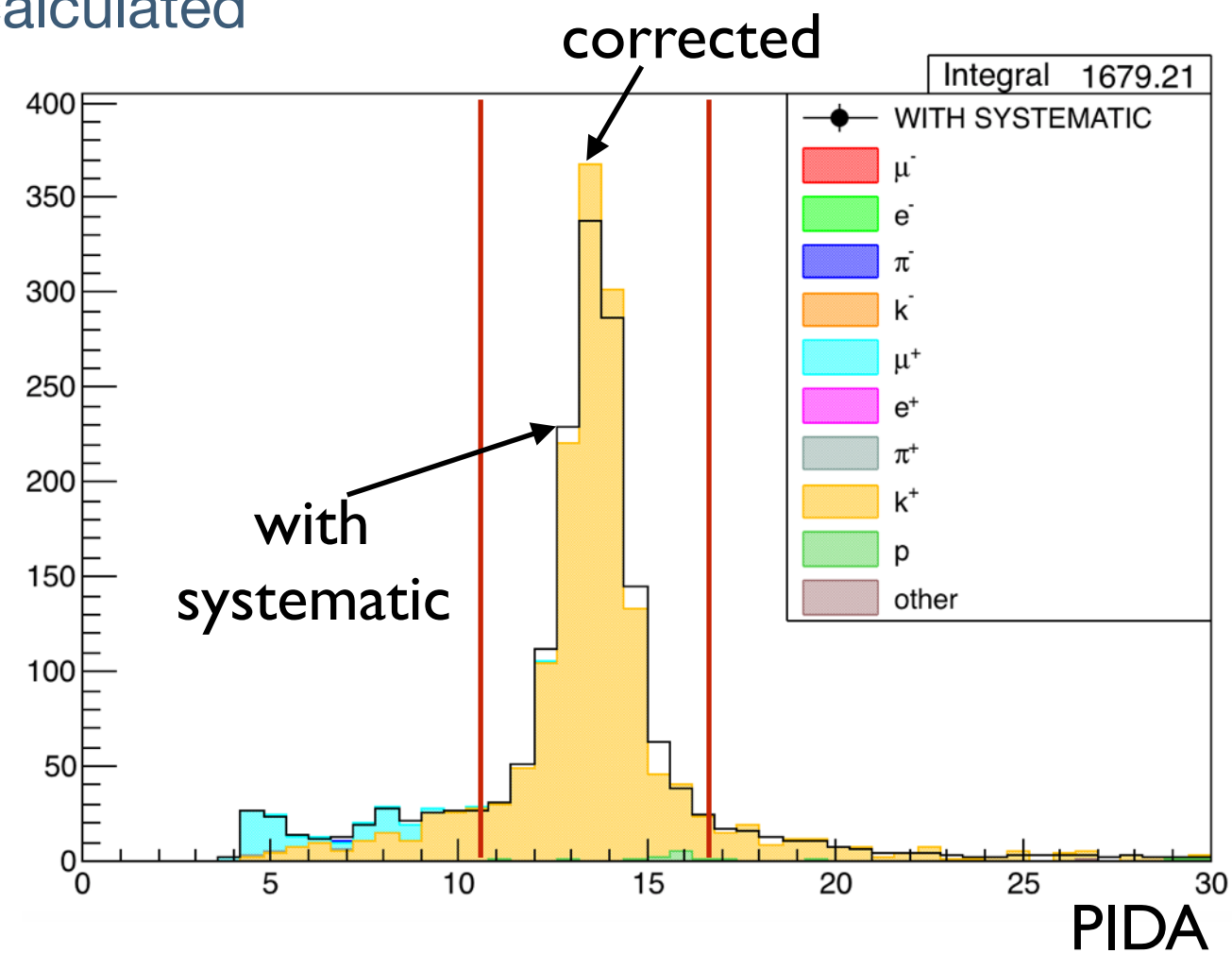
- full systematic propagation functionality is one of the main HighLAND benefits
- Systematic are propagated numerically by multiple throws (toy experiments)
- Two type of systematics:
  - **Event Variations**: Modify the input data
  - **Event Weights**: Just a global weight for the event



dEdxVariation systematic

- The error on the correction is the systematic
- 100 toy experiments. Each toy applies a different correction factor to all hits. Then PIDA is recalculated

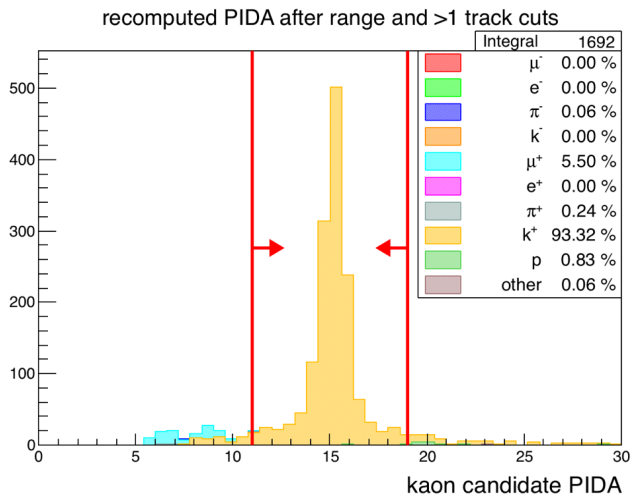
bins of PDG		correction	error on correction
10	12	0.95	0.02
12	14	0.98	0.02
320	322	0.9	0.02
2211	2213	0.8	0.02
210	212	0.9	0.02



The different levels

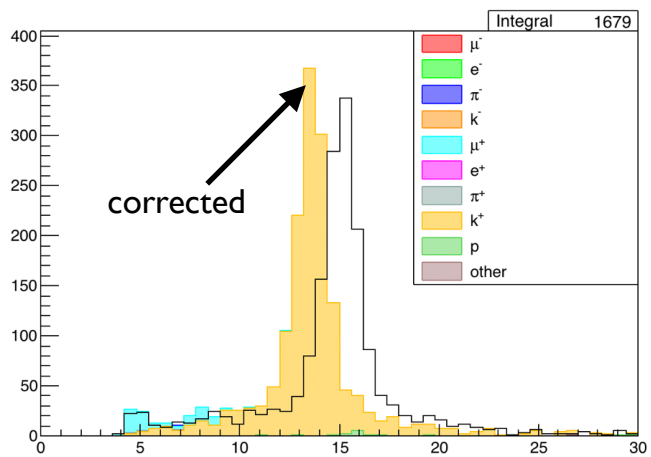
- The three levels are available in the HighLAND output file

raw PIDA



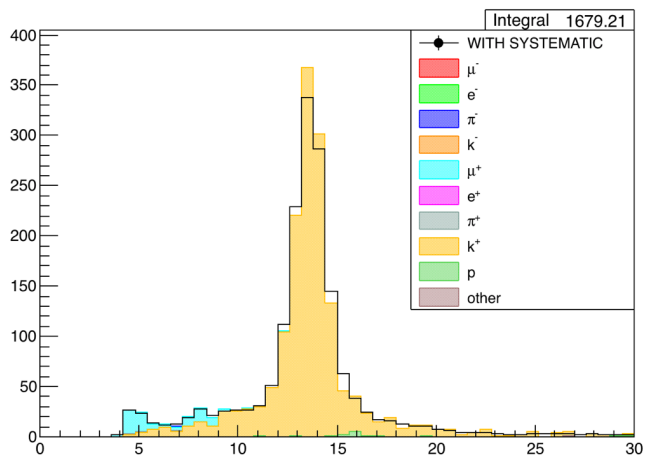
single value for each event

corrected PIDA



single value for each event

systematic propagated PIDA

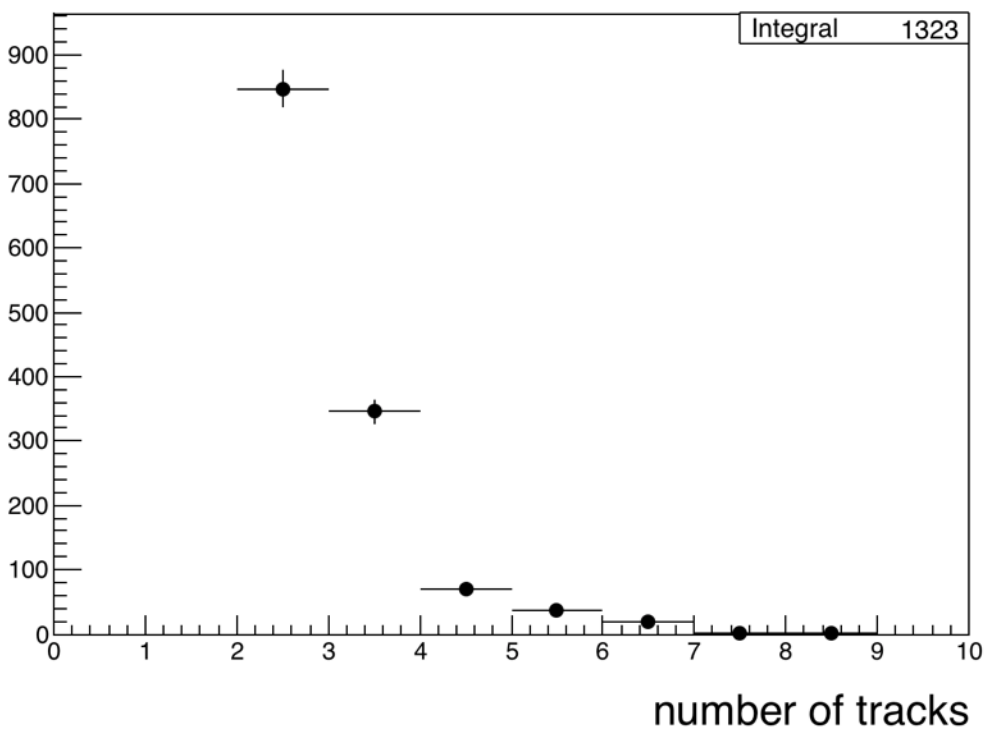


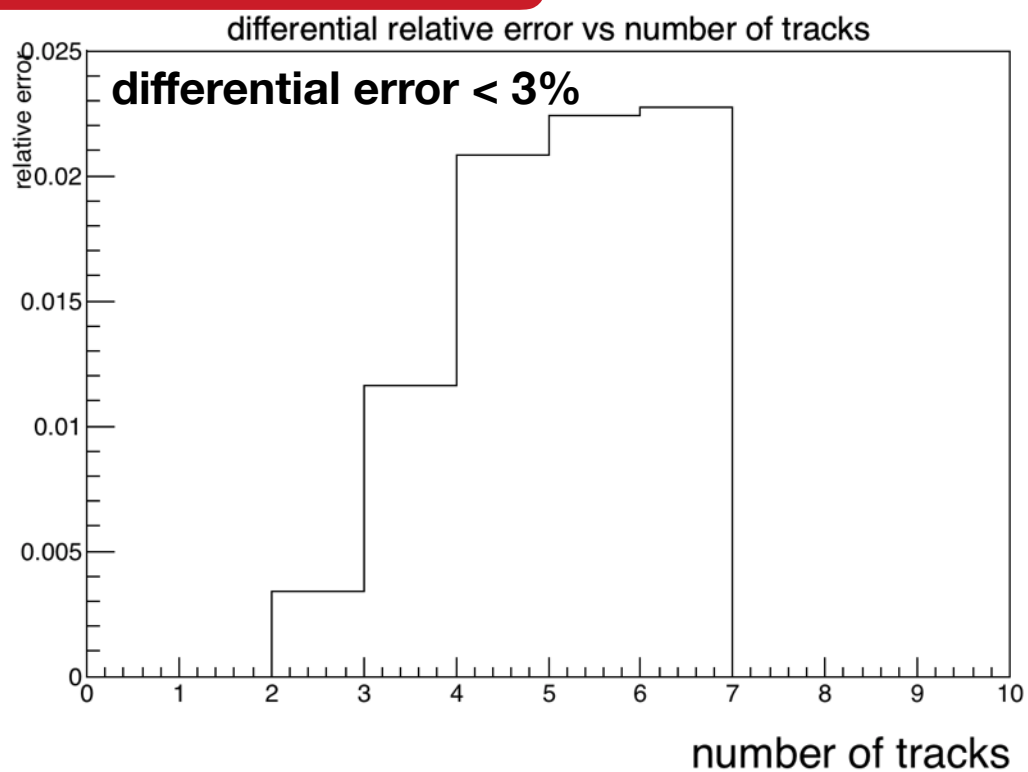
100 values for each event

Effect on the selection

- When the PIDA cut is applied the dEdx systematic has an effect on the number of selected events
  - Integrated: **0.3%**
  - Differential: **< 3%**

for events passing all cuts, including PIDA

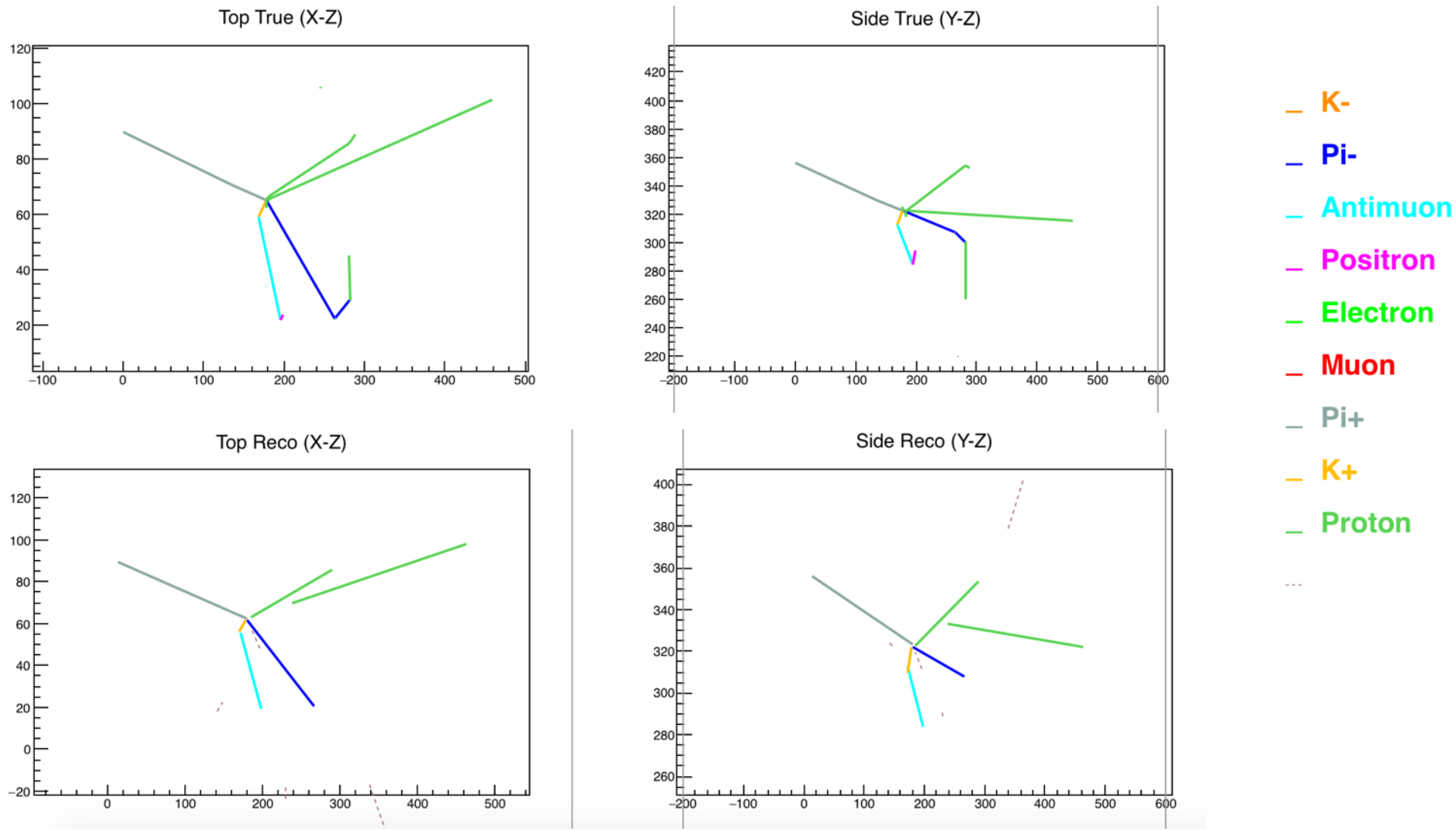




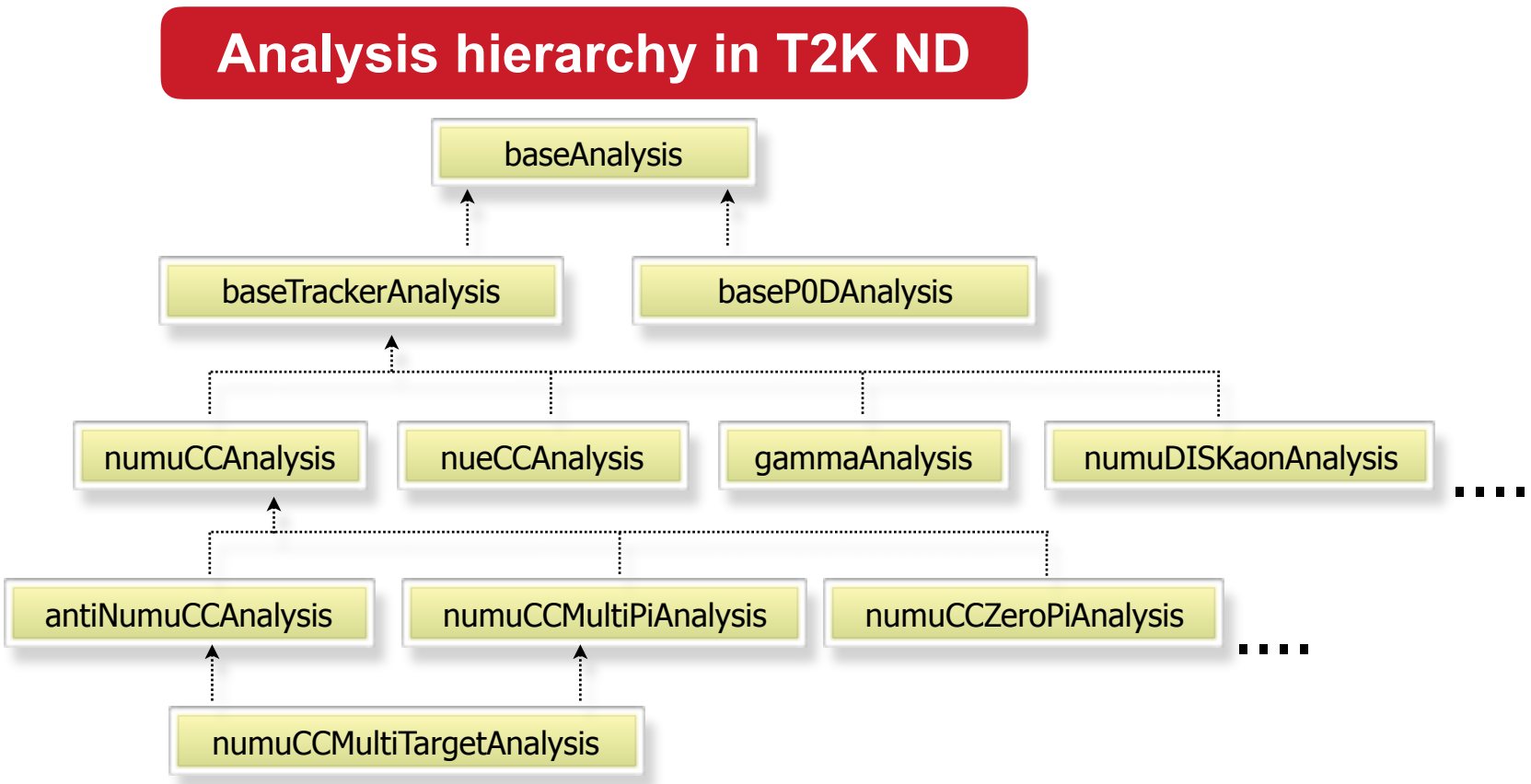
HighLAND event display

Secondary kaons from hadronic interactions A. Izmaylov

https://indico.fnal.gov/event/10641/session/12/contribution/81/material/slides/0.pdf



A hierarchy of analyses



only few packages shown here (>20)

- In most cases packages down in the hierarchy perform selections that are subsamples of the packages above
- But this is not mandatory, you can just use functionality from another package

The path forward

Next steps

- The next step should be to export an existing ProtoDUNE-SP analysis from LArSoft to HighLAND and try to reproduce the results
    - Using data and MCC10
- Try to solve the issue related to root6 cint
- Migrate to cmake
- Improve documentation

backup

Systematics

- Systematics are propagated numerically using toy- experiments (pseudo-experiments or virtual analyses)
- Each toy-experiment is defined by a set of random throws (one for each systematic parameter)
- The covariance of the number of events selected in a given bin is computed in the usual way:

$$C_{ij} = \frac{1}{N_{toys}} \sum_{t=1}^{N_{toys}} (N_i^t - \overline{N}_i)(N_j^t - \overline{N}_j)$$

events in bin i for toy t

$$N_i^t = \sum_{e=1}^{N_{events}} W_{e,i}^t$$

average over toys

$$\overline{N}_i = \frac{1}{N_{toys}} \sum_{t=1}^{N_{toys}} N_i^t$$